Network Architecture Search and Expert Designed CNNs for Multi-target Concrete Defect Detection

Zhiye Wang, Haixin Shi, Guosheng Feng
Department of Computer Science, EPFL, Switzerland

***Abstract*—In this project, we focus at the problem of bridge crack detection in CODEBRIM dataset for assessing the structural integrity in multi-target scenario. We compare two branches of convolutional neural networks (CNNs): expert-designed complex CNNs (e.g. VGG, ResNet, AlexNet) and automatically generated lightweight CNNs from network architecture search (NAS). For the latter, we introduce three NAS methods driven by different data source: in-domain models searched in CODEBRIM dataset, transferred models searched in ImageNet dataset and data-free zero-shot models. Further, we show that cross-domain transfer learning greatly boost our models' performance (ZenNAS-1 model from 69.5% to 75.6%), which surpassed all expert-designed models with much fewer parameters (134.28M from VGG-D to 5.43M from ZenNAS-1). We validate such performance gain via studying the attention distribution of convolutional filters using Grad-CAM on image samples of different cracks.**

I. Introduction

Concrete damage detection is a challenging task with following facts: (1). Real-world concrete defects are often overlapping with each other, as in Figure 1 (i.e. multi-label data). (2). Concrete defects data is hard to collect and often suffers from label imbalance problem. (3). Traditional expert-designed CNN models are heavy and hard to deploy in embedded vision applications. And designing CNNs adapted for new dataset often requires human involvement and expert knowledge.

Then, [1] firstly applied NAS methods for searching for powerful architectures in CODEBRIM crack detection dataset with two different NAS methods, and achieved best multi-target accuracy via MetaQNN method [2]. However, the MetaQNN is computational expensive that requires hundreds of GPU hours.[1]

Based on the above facts, in this project, we extend the work of [1], and explore and compare NAS and human expert designed CNNs for the multi-target classification task in CODEBRIM dataset. For the NAS methods, we focus on lightweight NAS algorithms as limited by the GPU resources. We firstly reproduce the ENAS results in [1]. Then, instead of using MetaQNN as the original paper, we apply two light NAS algorithms driven by different data sources: EfficientNet [3] searched on ImageNet dataset [4] and ZenNAS [5] searched in a data-free (i.e. zero shot) manner. Highlight and main contributions of this project as listed as follows:

- We evaluate and compare expert designed and NAS generated CNN architectures for the multi-target concrete defect classification task, and obtain highly competitive result using ZenNAS with much less parameters compared to the expert-designed models.
- Further, We show that cross-domain transfer learning could greatly boost the model performance, under which the ZenNAS model achieves best multi-target accuracy and surpassed the best result from MetaQNN in the original CODEBRIM paper [1] (from 72.2% to 75.6%)[2].
- We validate the performance gain using Grad-CAM to inspect attention pattern of last few convolutional layers, which shows that our transferred models' attention is better aligned with the defect area.

[1]the original MetaQNN work searched for 8-10 days by parallel training using 10 NVIDIA GPUs.

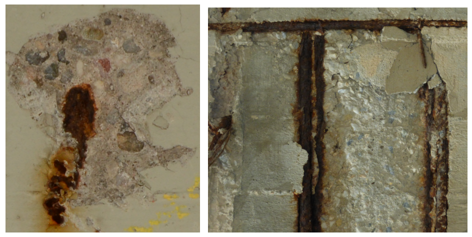


Figure 1: Two concrete image samples from CODEBRIM dataset, each have multiple overlapped defects. Left: Concrete with spallation, exposed bars and corrosion stain; Right: Concrete with spallation and exposed bars.

II. CODEBRIM Dataset

We use COncrete Defect BRidge IMage Dataset [1] which provides multi-target concrete defect images collected by camera and UAVs from different bridges. The dataset consists of six classes among 7736 images, i.e., cracks (2185), spalling (2208), efflorescence (1608), exposed bars (543), corrosion stain (1215) and background (1263). Each concrete image could contain multiple overlapped defects as shown in Figure 1. Therefore, following [1], we treat this task as a multi-target multi-class image classification problem, while also inspecting the single-target metrics in our experiments.

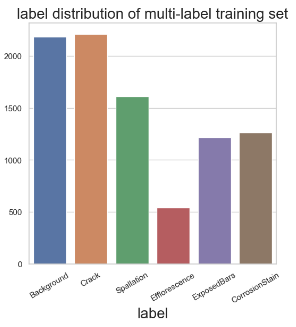
Meanwhile, we explore several perspective of this dataset, and found that the label distribution is imbalanced as shown in Figure 2a, and the resolutions of images are highly dispersed as in Figure 2b, due to the large variation of bounding boxes. To address the first problem, we explore several data balance and augmentation method such as focal-loss and over-sampling. And for the second problem, we set the patch-size as one of our hyper-parameters. Then, we first resize the image size to 256 then randomly crop it to different patch-sizes. Details will be discussed in section IV-A.

III. Models

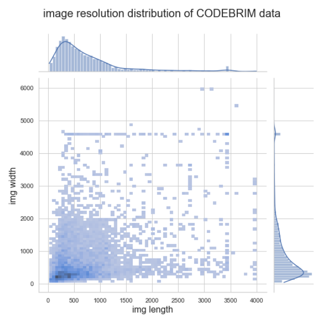
A. NAS-designed models

We use neural architecture search (NAS) methods for automatically generating dataset-adapted lightweight models. In a word,

[2]We maintain this as both our report and the original paper use grid-search for determining hyper-parameters, while ZenNAS model is worse than MetaQNN before transfer learning.



(a) Label disbribution



(b) Image resolution distribution

Figure 2: Data label and resolution distribution.

NAS aims at discovering the models constructed in the pre-defined search space with highest validation reward, and use search algorithm (e.g., policy gradient, reinforcement learning or genetic algorithm) for guiding the search process. We formalize the overall object of NAS as:

$$M^* = argmax_{M \in S}\ \mathbb{E}[R_V(M)] \tag{1}$$

where S is the pre-defined search space consists of huge amount of candidate architectures. R_V is the validation reward function, which depends on the choose of metrics and validation data source (e.g., multi-target accuracy in crack classification or zen-score for data-free search). Here we choose three NAS methods, i.e., ENAS, EfficientNet, ZenNAS, each with different search space and validation reward metric:

- **ENAS** [6]: ENAS sample CNN architectures in a directed acyclic graph (i.e., search space S), where each node represents a set of possible feature operations. ENAS uses LSTM to perform policy gradient for heuristically guiding the search process, while the sampling weight is shared among the whole DAG.
- **EfficientNet** [3]: EfficientNet uses multi-objective neural architecture search using a factorized search space as [7]. The search algorithm follows Proximal Policy Optimization [8].
- **ZenNAS** [5]: ZenNAS uses evolutionary algorithm to search architectures in a data-free manner, by using a zero-shot index dubbed Zen-Score to rank the architectures.

B. Expert-designed Models

Except the NAS designed CNNs, we choose several expert designed ones as our baselines, incuding DenseNet [9], VGG [10] variants including VGG-A and VGG-D model that contains 11 and 16 convolutional layers respectively, AlexNet [11] and ResNet variations including Resnet-50 and Resnet-101 (the suffix number refers to model depth). We adjust the input layer and last linear layer to adapt models to our dataset.

IV. Experiments

A. Experimental Setup

Training Settings: We first give our settings for training and evaluating models in CODEBRIM dataset. Firstly, for the multi-label nature of our data, we apply a Sigmoid function to map our

		Multi-target accuracy [%] of different LR Schedule					
		$[10^{-1}, 10^{-5}]$		$[5 \cdot 10^{-2}, 5 \cdot 10^{-4}]$		$\mathbf{[10^{-2}, 10^{-5}]}$	
		best val	bv-test	best val	bv-test	best val	bv-test
Batch Size	16	62.3	63.6	62.2	65.8	60.6	63.8
	32	61.0	64.9	60.2	64.2	62.8	64.7
	64	61.4	62.7	63.6	64.7	60.7	60.1
Weight decay	1e-3	30.7	34.7	51.3	56.0	62.8	63.8
	1e-4	64.1	64.7	61.7	65.2	62.7	63.4
	1e-5	61.9	63.0	61.2	60.1	63.1	64.6
Patch Size	32	19.5	14.6	13.5	8.7	55.7	58.7
	64	59.6	62.7	61.4	61.9	61.4	59.3
	128	59.7	62.3	63.0	63.0	62.0	62.7
	224	62.3	63.0	62.2	63.3	63.5	64.9

Table I: Hyper-parameter grid-search result. We apply the EfficietNet-b0 model as baseline. Models are trained from scratch without warm-restart following the settings described in Section IV-C

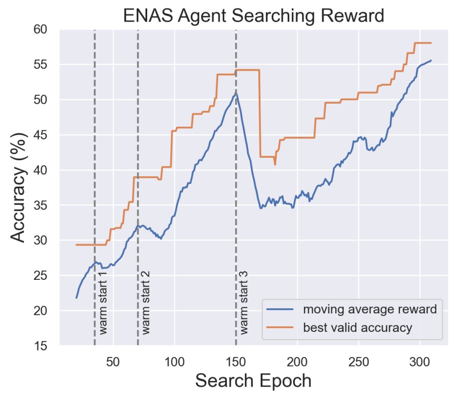


Figure 3: ENAS model evolution process, we show the moving average reward (window size=10) of multi-target validation accuracy on CODEBRIM dataset, with three warm restart points among 310 epochs. We extract the top-3 models encountered during the search, denoted as ENAS-1, ENAS-2, ENAS-3.

models' output followed by a Binary Cross Entropy loss for each class. The threshold of prediction is set to 0.5. Here we also use the SGD with warm-restart (SGDWR) optimizer with momentum 0.9. Other hyper-parameters such as weight decay and learning rate schedule are optimized using grid-search in section IV-C. After obtaining best hyper-parameters, we train each model following the same scheme: 200 epochs with 5 warm restart cycles, under which we see a stable model convergence.

Data Split: We reuse the data split in [1], where 150 unique defect examples per class are chosen for validation and test sets respectively.

Evaluation Metrics: All the models in our experiments is chosen according to the validation performance (i.e., best val), then we evaluate the model using the weight from the same epoch of best validation to get the best testing accuracy (i.e., bv-test). By default we report the multi-target accuracy, while for model comparison we also conduct the experiments in single-target scenario.

B. Network Architecture Search

Here we describe our settings for each of the aforementioned NAS methods:

		Multi-target			Single-target			
Model Type	Model Name	best val	bv-train	bv-test	best val	bv-train	bv-test	Params[M]
Expert Designed	DenseNet	62.3	96.3	64.2	90.5	99.7	89.9	11.50
	VGG-A	67.0	95.8	70.1	91.6	99.7	91.3	128.79
	VGG-D	68.0	98.6	**71.0**	91.3	99.6	91.4	134.28
	AlexNet	61.2	95.8	63.8	89.0	99.1	89.1	57.02
	ResNet-50	62.0	97.6	62.7	89.7	99.5	89.5	23.5
	ResNet-101	61.2	84.7	61.7	89.3	96.0	89.3	42.5
NAS Designed	ENAS-1	68.0	80.4	68.1	-	-	-	4.68
	ENAS-2	66.4	87.5	67.3	-	-	-	6.05
	ENAS-3	63.5	80.4	67.8	-	-	-	**2.75**
	EfficientNet-b0†	65.4	89.4	68.4	90.8	97.6	90.8	4.02
	ZenNAS-1*	64.9	98.6	**69.5**	90.8	99.3	90.7	5.43
	ZenNAS-2*	63.0	95.5	63.0	89.6	99.8	88.9	9.27

Table II: Model evaluation results for multi-target and single-target scenarios. For each model, we report the best validation result, as well as the corresponding training and testing accuracy at the same epoch. Models with † are searched on ImageNet dataset while models with $*$ are obtained from data-free zero-shot search.

ENAS [6]. Here we reproduce the experiments in [1], and set the architecture with 7 convolutional layers. We search for 310 epochs with 5 SGDWR cycles using policy gradient. And the search process is guided by a LSTM controller trained using ADAM [12] with initial learning rate 10^{-3}. The search process is depicted in Figure 3.

EfficientNet [3]. Unlike ENAS, here we directly use the compounded EfficientNet-b0 model searched over ImageNet [4] dataset. We adjust the final fully connected layer to match the class number of our data.

ZenNAS [5]. We use the same search space as the original paper, and the evolutionary population size is set to 256, number of evolutionary iterations T is 96,000. For the search process (ZenScore evolution) please refer to our Appendix. The optimizer and learning rate is set as the same as ENAS, except the warm-restart mechanism which reduces the quality of sampled models during our experiments.

C. *Hyper-parameter tuning*

We treat batchsize, patchsize (i.e., input image size), learning rate schedule and weight decay as hyper-parameters and optimize them using simple grid search. Here we use EfficientNet-b0 as baseline model. To reduce the time cost, we only search for 100 epochs using SGD without warm-restart. When searching for one hyperparamter, the other three are fixed, thus we set the default batch size=16, lr schedure=$[10^{-2}, 10^{-5}]$, patch size=224 and weight decay=1e-5. The results are shown in Table I. It is worth noting that the model is in favor for bigger patch-size, and a 1e-3 weight decay leads to model under-fitting, while batch size and lr schedule do not affect the performance significantly.

D. *Data processing and label balance*

Here we address two problems in the original data: (1). The original image data has large variations in aspect ratio and resolution. And the training data only contains 7736 samples, which is insufficient. (2). The label is severely unbalanced.

Data processing and augmentation: For the diverse image resolutions, we found that most image fall in the resolution under 1000 as shown in Figure 2b. There we first resize the image to 256 and randomly crop it to 224 according to the grid-search result. Then,

Method Type	Method Name	best val	bv-train	bv-test
Geometric Distortionss	RandomFlip	65.1	89.4	68.4
	RandomRotation	64.1	76.2	67.4
	RandomCrop	66.9	76.6	67.2
Photometric Distortions	RandomPerspective	57.8	82.2	61.4
	GaussianBlur	61.0	80.8	63.8
	Sharpening	60.2	93.8	64.6
Mix Transform	RandomChoice	65.4	73.5	**68.8**

Table III: Comparison for different image augmentation methods.

Method Name	best val	bv-test	bv-test-single
Baseline	64.4	64.9	89.9
α-balanced BCE	63.0	63.8	89.7
Focal Loss	47.7	51.9	85.8
Oversampling	64.9	**68.4**	90.7

Table IV: Label balance experiments in CODEBRIM dataset. Here bv-test-single denotes single-target accuracy obtained at the same epoch of best val.

we set the over-sampling as basic data pre-process step, and explore 6 different image translation methods shown in Table III. We find that, in general, the dataset is in favor of geometric distortions than photometric distortions, while a mixed transformation combining all distortion methods with equal probability could slightly boost the performance.

Label balance: Here we explore two branches of methods: adjustment of loss function and oversampling. For the former, we try α-balanced loss which give weight $\alpha = \frac{1}{\#class}$ to different classes, and the focal loss [13] with $\gamma = 2$ that give more weights to misclassified/hard classes. For oversampling, we follow [1] and simply duplicate images from the minority class. The results in Table IV reveals that adjustment to loss functions is not helpful to the performance, and we attribute such fact to the loss function overly biased to the minor classes which causes the gradient descent process extremely unstable (see Appendix). And the oversampling method actually increase the model performance from unbalanced baseline.

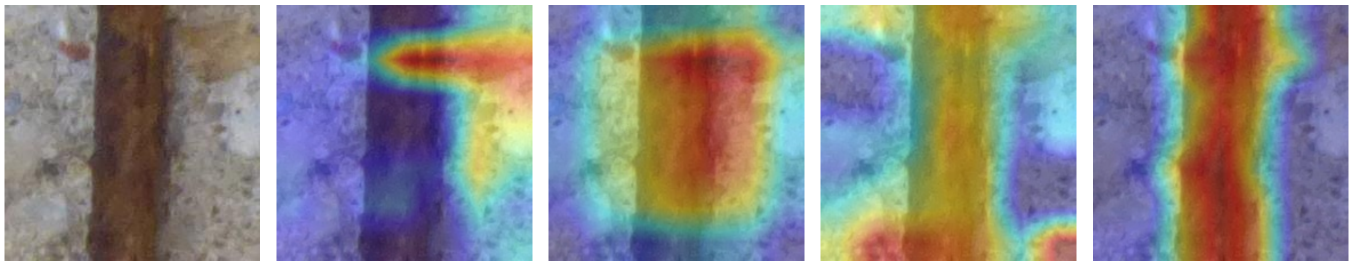

(a) Original image (b) ResNet W/O Pretrain (c) ResNet W/ Pretrain (d) ZenNAS-1 W/O Pretrain (e) ZenNAS-1 W/ Pretrain

Figure 4: Grad-CAM attention pattern in last convolutional layer. For more examples please refer to our Appendix.

Model	w/o pretrain		w/ pretrain		Params[M]
	best val	bv-test	best val	bv-test	
VGG-D	68.0	71.0	71.3	74.5	134.28
ResNet-50	62.0	62.7	73.2	75.2	23.5
EfficientNet-b0.	65.4	68.4	71.8	74.2	4.02
ZenNAS-1	64.9	69.5	70.6	**75.6**	5.43

Table V: Transfer learning experiment results. Pretrained models use weights from ImageNet dataset then retrain in CODEBRIM dataset.

E. Model Results

After building all aforementioned hyperparameter tunning and data processing piplines, we train all the models following the training settings in section IV-A. The results are listed in Table II. Here we report the metrics in multi-target and single-target scenarios. Note that the ENAS models and ZenNAS models with k variations (e.g., ENAS-1, ENAS-2) are obtained according to the best-k reward scores during the search process. Results show that expert-designed models tends to have more parameters than NAS-designed ones. The VGG-D model achieves best multi-target accuracy 71.0% with 134.28M parameters, while the ZenNAS-1 model reaches a competitive accuracy of 69.5% with only 5.43M parameters.

F. Transfer Learning

As the training data in CODEBRIM dataset is scarce, we believe some basic image patterns are not sufficiently learned. A direct way to solve this is transfer learning. Here we choose two expert-designed and two NAS-designed models, test their performance with or without pretrained weight from ImageNet dataset.[3] From results in Table V we observe that, transferred models' performance greatly increases (6.98 average gain in multi-target accuracy). It is interesting that all models with pretrained weight obtain similar performance, while ZenNAS-1 reaches best test accuracy 75.6%, which is higher than all the accuracies (best 72.2% from MetaQNN) reported in the CODEBRIM paper [1].

To further validate such performance gain, we use gradient-weighted class activation mapping (Grad-CAM) [14] to inspect CNN attention in the last convolutional filter. Grad-CAM method

[3]We use the ImageNet weights directly avaible in TorchHub, ZenNAS-ImageNet. Then we reinitialize the weight in the last classification layer and train the model in CODEBRIM dataset using same settings as before.

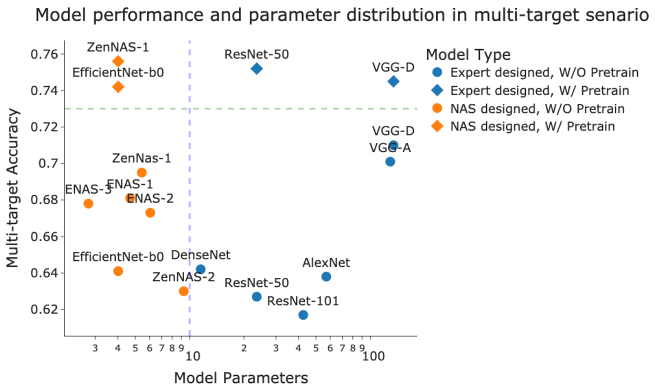


Figure 5: Overall model parameter-accuracy distribution of all our trained models.

compute the attention via gradients flowing into final conv layer to obtain a coarse localization map highlighting important regions. We inspect two models as ResNet-50 (expert-designed) and ZenNAS-1 (NAS-designed). The results are shown in 4, which clearly reveals that attention patterns of models with pretraining are better aligned with defect area. And the ZenNAS's pattern almost perfectly matched the defect region.

V. Conclusion and Future Work

Through this project, we apply NAS and expert designed CNNs to the multi-target concrete defect classification task. The overall model performance distribution is depicted in Figure 5, where NAS methods generate lightweight while accurate models, such as ENAS-1 and ZenNAS-1. Beside dealing with data and hyper-parameters, we find that transfer learning could significantly help with the model performance, where we study the attention pattern to validate this. For future work of this project, we believe more concrete dataset such as SDNET [15] and MCDS [16] could be used for a united in-domain transfer learning. And some fine-grained image classification models such as [17] may also helpful for the domain-specific concrete data. There is also vulnerability in our NAS models, as we have not fully explored the search space for more robust structures adapted for concrete data, and some advanced vision transformer models could also be helpful to address the relationships of multiple targets that overlapping in a single image.

References

[1] M. Mundt, S. Majumder, S. Murali, P. Panetsos, and V. Ramesh, "Meta-learning convolutional neural architectures for multi-target concrete defect classification with the concrete defect bridge image dataset," in *CVPR*. Computer Vision Foundation / IEEE, 2019, pp. 11 196–11 205.

[2] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *ICLR (Poster)*. OpenReview.net, 2017.

[3] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 97. PMLR, 2019, pp. 6105–6114.

[4] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*. IEEE Computer Society, 2009, pp. 248–255.

[5] M. Lin, P. Wang, Z. Sun, H. Chen, X. Sun, Q. Qian, H. Li, and R. Jin, "Zen-nas: A zero-shot NAS for high-performance deep image recognition," *CoRR*, vol. abs/2102.01063, 2021.

[6] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 80. PMLR, 2018, pp. 4092–4101.

[7] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *CVPR*. Computer Vision Foundation / IEEE, 2019, pp. 2820–2828.

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

[9] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," *CoRR*, vol. abs/1608.06993, 2016.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[11] A. Krizhevsky, "One weird trick for parallelizing convolutional neural networks," *CoRR*, vol. abs/1404.5997, 2014.

[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR (Poster)*, 2015.

[13] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *ICCV*. IEEE Computer Society, 2017, pp. 2999–3007.

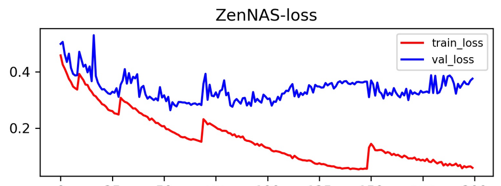
[14] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *ICCV*. IEEE Computer Society, 2017, pp. 618–626.

[15] C. Zhu, M. Zeng, and X. Huang, "Sdnet: Contextualized attention-based deep network for conversational question answering," *CoRR*, vol. abs/1812.03593, 2018.

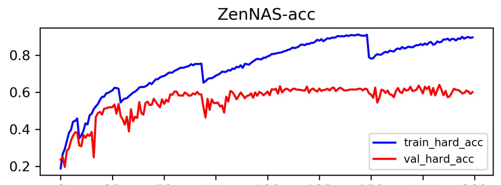
[16] P. Hüthwohl, R. Lu, and I. Brilakis, "Multi-classifier for reinforced concrete bridge defects," *Automation in Construction*, vol. 105, p. 102824, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092658051831269X

[17] F. A. Breiki, M. Ridzuan, and R. Grandhe, "Self-supervised learning for fine-grained image classification," *CoRR*, vol. abs/2107.13973, 2021.

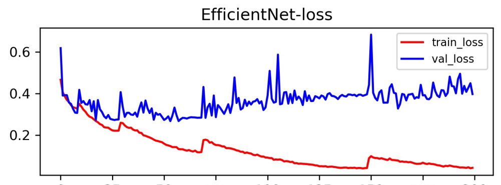
Appendix



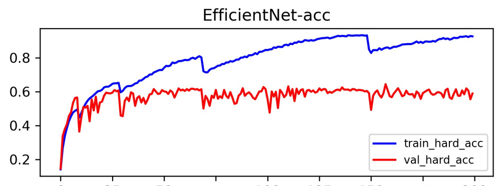
(a) ZenNAS loss



(b) ZenNAS multi-target accuracy

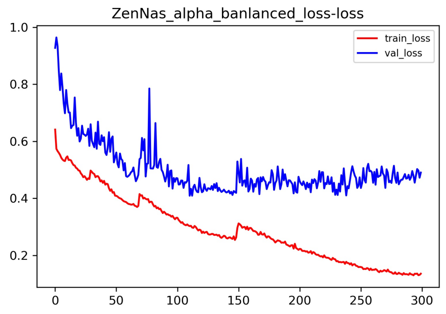


(c) EfficientNet loss

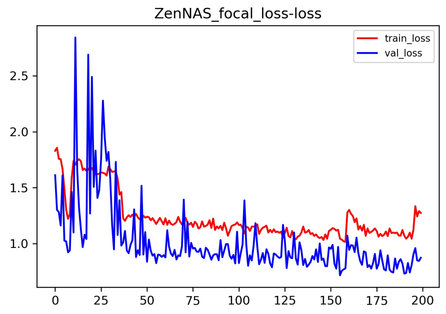


(d) EfficientNet multi-target accuracy

Figure 1: Training curve for ZenNAS and efficientNet with loss and multi-target accuracy.



(a) ZenNAS training curve with α-balanced loss



(b) ZenNAS training curve with focal loss

Figure 2: Training curve for ZenNAS equipped with α-balanced loss and focal loss.

I. Training epoch choice

We choose the epoch number for model training based on the observation of loss and accuracy curves. As depicted in Figure 1, we could observe that the model reaches convergence near 100 epochs which we choose for the epoch size for grid-search. By looking at the multi-target accuracy curve, we also see that model performance keeps going up after 100 epoch, and slightly increases after the final warm restart at the 150 epoch. Therefore, we choose 200 epoch for training the final model.

II. Inspect label-balanced loss

We assume the poor performance from α-balanced BCE loss and focal loss could be attribute to the fact that the model is over-biased to the minority class and punished by predicting towards majority classes. We plot the training curve in Figure 3, which reveals that the training process becomes unstable under these two adjusted loss functions. We tried to adjust the hyper-parameter for focal loss to be lower as $\gamma = 0.5$ or $\gamma = 1$ so as to reduce the model bias, after which the performance is close to model with the original loss (i.e., $\gamma = 0$). Same situation to α-balanced loss: when we decrease the α weight toward minority class, model performance become close to the one with the original loss (i.e. $\alpha = 1$ for all the classes).

III. ZenScore Evolution during ZenNAS search

For the limitation of main pages, we put the search process of ZenNAS here. Note that using ZenNAS search we do not

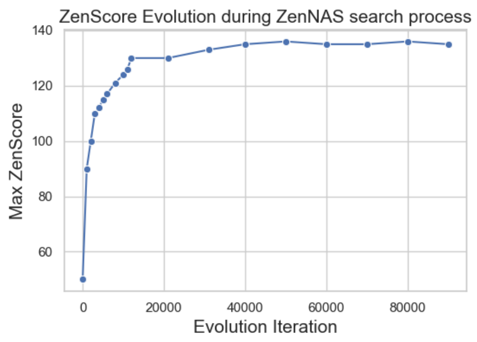


Figure 3: ZenNAS search process: best zenScore obtained at each iteration

apply CODEBRIM dataset, as ZenNAS searches in a data-free manner. The evaluation metrics used by ZenNAS is the ZenScore, which is obtained by re-scale the ϕ-score one more time by the product of BN layers' variance statistics, and the ϕ-score is defined as:

$$\phi(f) = log\mathbb{E}_{x,\theta}||\Delta_x f(x|\theta)||_F$$

which is the expected Gaussian complexity for a vanilla network $f(\cdot)$.

IV. Convolutional Attention for different classes

Here we present more examples of Grad-Cam attention pattern in ResNet and ZenNAS model in Figure 4. We arrange the images as same as the main report: each row from left to right : [original image, ResNet W/O pretrain, ResNet W/ pretrain, ZenNAS W/O pretrain, ZenNAS W/ pretrain]. Here we could see the resnet model with pretrained tend to have a ”rectangle”-like attention pattern located in the defect area, while the patterns of pretrained ZenNAS model are more flexible and could better align with the defect region.

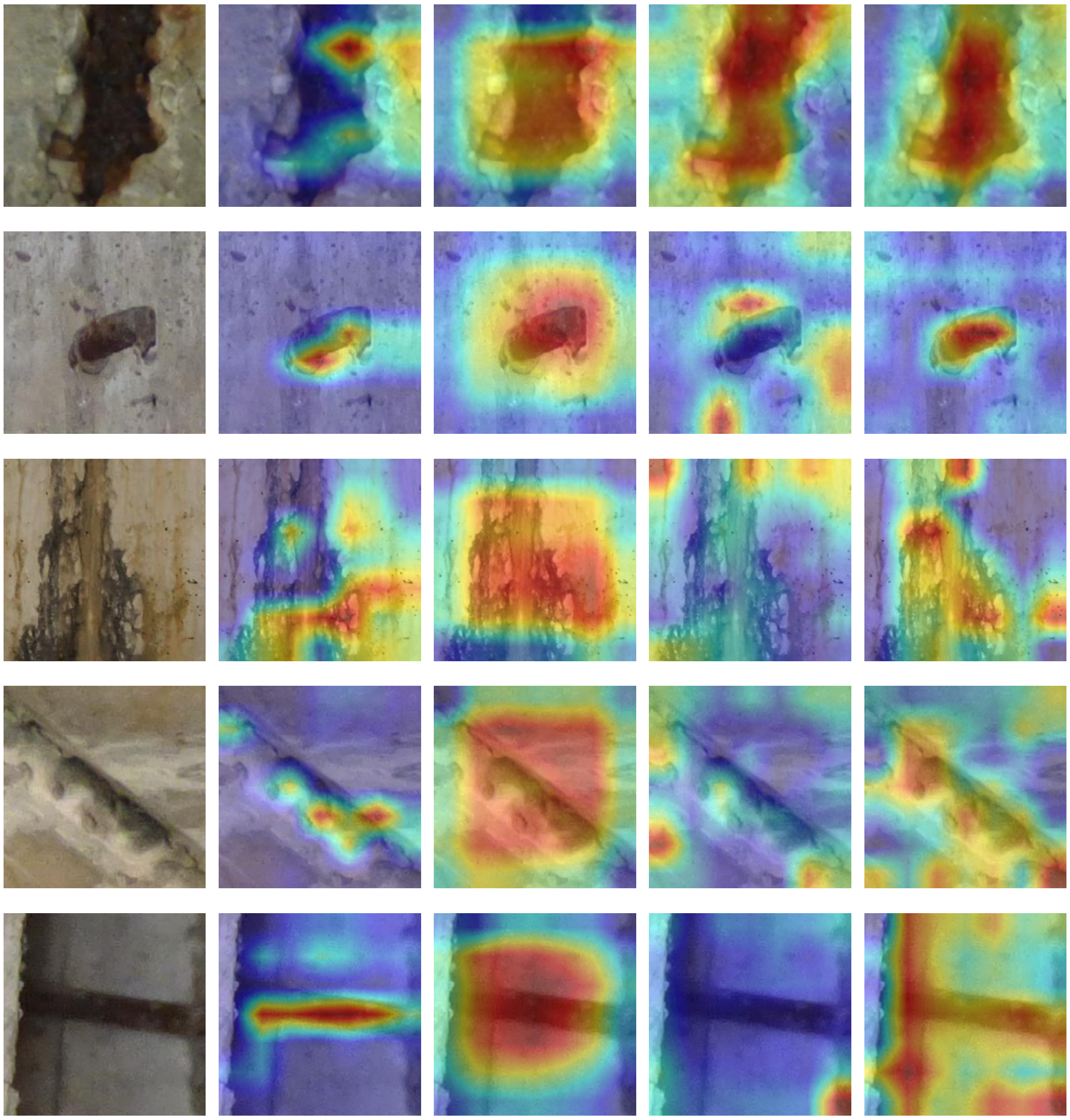


Figure 4: More examples on model attention pattern in the last convolutional layer. Labels for rows from top to bottom: [Spallation, ExposedBars, CorrosionStain], [ExposedBars], [Efflorescence, CorrosionStain], [Efflorescence], [ExposedBars]. Images in each row from left to right: original image, ResNet W/O pretrain, ResNet W/ pretrain, ZenNAS W/O pretrain, ZenNAS W/ pretrain.

.